REMARKS

Jiang

Jiang[1] draws attention to a difficulty with storage systems that use multiple controllers, namely that when one controller receives a request to write data, that controller writes all the data, even if other controllers are idle. *Jiang* discloses a method for enabling two or more controllers to cooperate, with each controller writing a portion of the data.

Of course, if all the controllers are trying to write different portions of the data to the same disk, it does little good for them to work in parallel, since only one of them can write to the disk at any time. However, in RAID systems, each controller can write one stripe. Since stripes are written to separate disks, the controllers can work in parallel so that different portions of the data are written concurrently.[2]

As a threshold matter, it is apparent that *Jiang* discloses methods for managing a RAID system, and has in fact little to do with data mirroring.

SECTION 102 REJECTION OF CLAIMS 1 AND 14

Applicant cancels claims 1 and 14 and rewrites claims 2 and 15 in independent form.

SECTION 103 REJECTION OF CLAIM 2

Writing to temporary storage location

As best understood, the Examiner regards claim 2's step of

> *"writing first and second portions of the data to respective first and second slots within the temporary storage location"*

to be disclosed by *Jiang*'s paragraphs 11-12.

In the cited paragraphs, *Jiang* states that

---

[1] *Jiang*, US 2003/0212860.
[2] *Jiang*, paragraphs 13-14.

> *"when a data controller A...receives a data write request..., then the data controller A writes a portion of the data to be written into the storage resource"*[3]

This raises the question: "what does *Jiang* mean by 'storage resource'"? The answer to this lies in FIG. 1, in which one plainly sees a plurality of disks collectively labeled "storage resource **40**."

In the context of a RAID system, which is what *Jiang* discloses, paragraphs 11-12 clearly refer to having one controller write a stripe to one disk and having another controller write another stripe to a different disk.

Applicant submits that the disk array **40** cannot reasonably be regarded as a "temporary storage location."

Buffering a mirror request

The Examiner apparently considers the APA[4] step of sending the mirror request (step **114**) to correspond to buffering a mirror request.

Step **114** is the step of sending the mirror request to the mirror queue.[5] Execution of step **114** marks a point of no return. Once step **114** is carried out, mirroring proceeds to completion, even if it is later discovered that the data specified by the mirror request is invalid. This is different from "buffering" the mirror request. When a mirror request is buffered, it can still be deleted.

Deleting a mirror request

The Examiner appears to consider the step of "deleting the first and second mirror requests" to be suggested by the APA step **128** of FIG. 2.

Step **128** is clearly labeled "Notify host of failed write." This is not the same thing as deleting a mirror request. This is simply sending a message notifying a host of a failed write.

---

[3] *Jiang*, paragraph 11.
[4] *Admitted Prior Art.*
[5] *Specification*, page 8, second paragraph.

The Examiner appears to be assuming that somewhere along the line, a mirror request is deleted. In fact, when the flow of control is such that step **128** is executed, there is no mirror request to delete.

Summary

Accordingly, even if one were to somehow combine *Jiang* with the admitted prior art, the result would fail to meet the limitations of claim 2.

Claims 3-5 include the limitations of claim 2 and are therefore patentable for at least the same reasons.

SECTION 103 REJECTION OF CLAIM 7

"Buffering" and "sending" are different

As best understood from the remarks on page 4 of the office action, the Examiner considers step **114** in FIG. 2 to correspond to *both* buffering *and* sending a mirror request. Thus, in the Examiner's view, there is no distinction between buffering a mirror request and sending a mirror request for execution. Particularly revealing is the Examiner's statement, concerning the buffering step, that "Applicant's admitted prior art discloses a mirror queue."[6]

In fact, buffering the mirror request need not involve the mirror queue at all. It is the step of *sending* the mirror request for execution that involves the mirror queue.

The introduction of a buffering step enables mirror requests to be *generated but withheld* from the mirror queue until such time as those mirror requests are known to correspond to valid data. Once this occurs, the buffered mirror requests are all released to the mirror queue together. If the data is invalid, the buffered mirror requests are all deleted. This procedure avoids the problem of having released a mirror request to the queue only to discover later that the data being mirrored by that request is invalid.

---

[6] *Office Action*, page 4.

Deleting a mirror request

The Examiner appears to consider the step of "deleting the first and second mirror requests" to be suggested by step **128** of FIG. 2. However, step **128** is to "Notify host of failed write." This is not the same thing as deleting a mirror request. In fact, when the flow of control is such that step **128** is executed, there is no mirror request to delete.

First and second portions of data

With regard to the proposed combination with *Jiang*, Applicant incorporates herein the arguments made in connection with the rejection of claim 2.

SECTION 103 REJECTION OF CLAIM 8

Applicant amends independent claim 8 to clearly recite the limitation that more than one mirror request is needed to copy the data-to-be-mirrored. As recited in claim 8, the plural mirror requests are *buffered*. This allows data to be inspected for invalidity before the irrevocable commitment associated with releasing a mirror request to the mirror queue.

APA FIG. 2 fails to disclose "buffering a plurality of mirror requests." In FIG. 2, mirror requests are placed on the mirror queue in step **114**. Thus, once step **114** is executed, mirroring proceeds to completion, even if it is later discovered that the data being mirrored is invalid.

The fact that the mirror request is placed in a queue, and that some time may elapse before mirroring actually occurs, is not definitive of buffering. By way of analogy, a stream of machine gun bullets in-flight is in effect a "queue" of bullets waiting to reach their target. However, nobody would seriously say that these bullets are somehow "buffered" because they have not yet reached their target.

SECTION 103 REJECTION OF CLAIM 9

Claim 9 is similar to claim 7 but with the omission of steps executed in connection with valid data. Accordingly, Applicant reiterates the arguments concerning the deleting step and the buffering step as set forth in connection with claim 7.

SECTION 103 REJECTION OF CLAIM 15

Claims 15-18 recite limitations similar to claims 2-5. Accordingly, claims 15-18 are patentable for at least the same reasons discussed above in connection with claims 2-5.

SECTION 103 REJECTION OF CLAIM 10

Applicant draws attention to the last paragraph of claim 10, which recites the limitation that the mirror requests required for mirroring the data are *accumulated* in the holding pen before being sent to the mirror queue.

The Examiner appears to regard the "WR processing queue" in *Jiang* as corresponding to the claimed "holding pen."

According to *Jiang*, the WR processing queue for a particular controller accumulates write commands to be handled by that controller. *Jiang* is silent as to *when* those write commands are actually sent for execution. In particular, there is no suggestion that *Jiang* accumulates write commands before sending those write commands for execution.

In a queue of write commands, one would expect write commands to be released for execution as soon as possible. Thus, if a disk and the controller were both idle, and a write command were waiting in the WR processing queue, one would expect the controller to release the write command for prompt execution. There would be no reason for the controller to *wait* until another write command has accumulated in the WR processing queue. After all, doing so would simply waste time.

In contrast, the claimed invention recites a holding pen that *accumulates* mirror requests *before* releasing them to the mirror queue. This is useful because once a mirror request is released to the mirror queue, it cannot easily be recalled. Thus, the release of a mirror request is, as a practical matter, *irrevocable*. The holding pen thus provides a way of avoiding the premature release of a mirror request that one might later regret.

Specifically, in some cases, data-to-be-mirrored requires two or more mirror requests. In such cases, there is a risk that a first mirror request, which writes only a portion of the data,

might be released to the mirror queue *before* one can determine that the data as a whole is valid. If the data-to-be-mirrored turns out to be invalid, there is no way to recall the mirror request. As a result, the data stored on the will also become invalid.

The invention avoids this difficulty by requiring that the mirror requests be accumulated before being released. This provides an opportunity for determining that the data to be mirrored is indeed valid. If the data-to-be-mirrored turns out to be valid, then all the mirror requests for that data can be released as a unit. If the data-to-be-mirrored turns out to be invalid, then all the mirror requests for the data can be deleted.

It is apparent therefore that *Jiang* fails to disclose a holding pen in which mirror requests are accumulated before being set to the mirror queue. Nor is such a holding pen disclosed by the admitted prior art. Accordingly, the proposed combination of the admitted prior art with *Jiang*, even if it were possible, would fail to yield the claimed invention.

SECTION 103 REJECTION OF CLAIM 12

Applicant amends claim 12 to be more consistent with the specification, which refers to the global memory **22** on page 5. The global memory **22** is also referred to as a "cache" memory, since it is used for temporary storage of data before the data is written to disk. However, the term "global" more clearly distinguishes this memory from the local memory recited in claim 11.

The Examiner suggests that claim 12 is obvious because a queue for accumulating mirror requests can be maintained in *any* memory having sufficient space.

The Examiner's statement however overlooks the memory architecture implicit in the word "global." A global memory, as taught in the specification, is accessible by different host adaptors and remote adaptors.

Jiang discloses cache memories **24, 34** and control memories **20, 30**. However, these are accessible only to their respective controllers **20, 30**. *Jiang* fails to disclose any structure that could reasonably be called a "global memory."

SUMMARY

Now pending in this application are claims 2-13 and 15-19. Of these, claims 2, 7-10, and 14 are independent. Enclosed is a check for the Petition for Extension of Time fee. No additional fees are believed to be due in connection with the filing of this response. However, to the extent fees are due, or if a refund is forthcoming, please adjust our deposit account 06-1050, referencing attorney docket "07072-948001."

Respectfully submitted,

Date: 3/9/2006

Faustino A. Lichauco
Reg. No. 41,942

Fish & Richardson P.C.
225 Franklin Street
Boston, MA 02110
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

21269583.doc